Lecture 14: Memory management

- Hashed paging
- Abusing permission bits
- Page replacement
- (time permitting) Thrashing
IPT (Hashed paging)

- table with entry per frame containing PID, page
  in that frame.
  - if log addr space huge, much smaller table.

- Need to search to find a page

\[
\text{access } P_1, \text{ page } 13
\]

\[
\text{P}_3 \rightarrow \text{P}_2 \rightarrow \text{P}_1
\]

- Each page can only be in a small # of potential sets.
- Only need to look in those places.

- Can have collisions
  - extremely unlikely all of the hashes for a small # of different pages to all collide.
  - unlikely to have a lot of competition for a small # of possible frame
  5s.
TLB protection bits

- Segmentation: Try to write to code section? Segmentation create a small read MUX pointer
- Invalid create page full of NULLS (e.g. 0 or -1). Page
- When created, mark as "invalid" in PT
- If process tries to read a "invalid" (any fault), at that point, copy a page
- Copy on write pages: Limit to copy a page

![Diagram of shared memory between processes]

**Debugger watch points**

- Controlling process request a signal if controlled process updates a page
- Debugger program share memory: debugger pause program to inspect new value of variables.
Page replacement:

- more logical pages than physical memory.
- extra pages need to be swapped out (to disk)
  [extremely slow!]

- When you access a page (not in memory), need to kick out (swap out)
  a page that is → free up a frame
  for P.

Need to decide what to evict.
Belady's anomaly:
- With some page replacement algorithms, can actually get worse performance with more memory, more page faults.